Towards an Application Framework for Automated Planning and Scheduling

Alex Fukunaga, Gregg Rabideau,
Steve Chien, David Yan
Jet Propulsion Laboratory, MS 525-3660
California Institute of Technology
4800 Oak Grove Drive
Pasadena, CA 91109-8099
(818)306-6157
alex.fukunaga@jpl.nasa.gov, gregg.rabideau@jpl.nasa.gov,
steve.chien@jpl.nasa.gov, david.yan@jpl.nasa.gov

Abstract- A number of successful applications of automated planning and scheduling applications to spacecraft operations have recently been reported in the literature. However, these applications have been one-of-a-kind applications that required a substantial amount of development effort. In this paper, we describe ASPEN, a modular, reconfigurable application frame.wolk which is capable of supporting a wide variety of planning and scheduling applications. We describe the architecture of ASPEN, as well as a number of current spacecraft control/operations applications in progress.

TABLE OF CONTENTS

1. INTRODUCTION

Automated planning/scheduling technologies have great promise in reducing operations cost and increasing the autonomy of aerospace systems. Planning[1] is the selection and sequencing of activities such that they achieve one or more goals and satisfy a set of domain constraints. Scheduling selects among alternative plans and assigns resources and times for each activity so that the assignments obey the. temporal restrictions of activities and the capacity limitations of a set of shared resources. In addition, scheduling is an optimization task in which metrics such as tardiness and makespan are minimized. Scheduling is a classical combinatorial problem that has long been studied by researchers in operations research. While traditional OR approaches (cf. [7]) have focused on optimal solutions for highly restricted classes of problems, there has been much recent interest in the heuristic, constraint-base.d approaches that are applicable to practical domains.

Traditionally, the problems of planning and scheduling have been studied separately. Recently, approaches that integrate. both the planning and scheduling process together under a unifying framework have been developed. Some recent aerospace

[1] We take these definitions of planning/scheduling from [4].

applications of hybrid planner/schec lulers include [13,17, 16].

Although the benefits of applying planning/scheduling technology can be significant, developing real-world, planning/sched uling systems is often an extremely time-consuming task. Modeling a complex domain requires an expressive modeling languag e, as well as data structu res that represent the constraints expressed in the domain model. In addition, complex, data structures and algorithms that s11])])01 t incremental modifications Io candidate plans/schedules need to be des igned and implemented.

In order to enable the rapid development of au tomated scheduling systems foJ" NASA applications, we have developed AS]'] {N (Automated Scheduling and Planning I {Nvironment), a reusable, configurable, generic planning/sch eduling application framework. An application framewoi k [15] is a class library (i.e., a i eusable set of software components) that provides the functionality of the components found in prototypical instances of a particular application domain.

ASPEN is an object-m iented system that provides a reusal)le set of software components that implements the el ements commonly found i]] complex planning/sch eduling systems. These include:

- An expressive constraint modeling language to allow the user to naturally define the application domain;
- A constraint management system for representing and maintai ning spacecraft operability and re source constraints, as well as activity requirements;
- A temporal reasoning system for expressing and maintaining temporal constraints; and
- A graphical interface for visualizing plans/schedules (for use in mixed-initiative systems in which the problem solving process is interactive).

ASPEN is currently being utilized in the development of an automated planner/scheduler for commanding a naval communications satellite, as well as a scheduler for the ground maintenance for the Reusable Launch Vehicle. In Section 2, we discuss application frameworks, and their applicability to planning/scheduling systems. Section 3 describes the architecture of the ASPEN. Section 4 describes some current applications of the ASPEN frame.work, includ ing g rou nd maintenance scheduling for the Reusable Launch Vehicle, as well as operations planning/schedu ling for two autonomous satellites. Finally, Section 5 describes related work.

2. APPLICATION FRAMEWORKS

Object-oriented software development techniques make it possible to develop generic applications for specific domains. These application frameworks consist of a class library forming a frame that has to be customized for a specific application. The concept of application frameworks was pioneered in the domain of user interface application fi amewoi ks such as ET+ [20] or h4acApj)[21], which provide a reusable, blank applicat ion that implements much of a given useri nterface lo(k-aml-feel standai d. 1 'or any particular application, a programmer can concenti ate on implementing the application-specific parts.

We first discuss some aspects of class libi aries before defining the term application fi amework. Compared to conventional rout ine librai its, class libi aries are hiei archical, with the most general class at the top of the hierarchy tree (if single inheritance is used).

This hierarchical organization helps to reduce the complexity of a library. An important print.i])1c behind the design of a class hierarchy is that the common behavior of classes is factored out into their superclasses.

Classes which factor out common behavior of other classes typically contain some methods that cannot be implemented. Any class that contains one or more "empty" methods (i.e., methods with some kind of dummy implementation) is termed an *abstract class*. It does not make sense to generate instances of them. Nevertheless, abstract cl asses may also contain methods that can already be implemented in advance for all subclasses.

The most important aspect of abstract classes is that they form the basis of *extensible* and *reusable* software systems: it is possible to realize whole software systems using only abstract classes, i.e.., the protocol supported by them (we define the term "*protocol* of a class" as all the. methods and instance variables provided by a class). If subclasses of abstract c.lasses are added to the class libary, these software systems need not be changed. 'I'l Icy also work with the objects of new subclasses of time abstract classes (on which other software systems ar e based), since these objects support at least the protocol (though implemented in a specific manner) defined in their (abstract) superclasses. The methods of abstract classes are dynamically bound, so that the corresponding methods of the objects whit.11 are instances of the new c.lasses are called at run time.

New subclasses of abstract Classes call reuse all the code. that was already implemented in their superclasses. Class libraries are called *application frameworks* if t t ıey apply the ideas presented above in order to provide a software system which is a generic application for a specific domain. Classes comprising a scheduling algorithm, together with all the abstract classes it rc.lies on, form a scheduling application framework. Applications base.d on such an application framework are built by customizing its abstract and concrete classes, Thus, a given framework already anticipates much an application's design which is reused in all applications based on the classes of that application framework. This implies a significant reduction in the amount of code necessary to implement successive systems.

3. ASPEN: A FRAMEWORK FOR PLANNING/SCHEDULING SYSTEMS

Note that the development of an application framework for a particular domain implies a standardized approach to implementing systems for that domain, and a commitment by the fr amewor k dev eloper to support applications that confor m to that stan dard ized approach. It is impr act ical to develop a framework to support *all* viable. approaches to planning and scheduling. Numerous, widely divergent approaches to planning and scheduling have been developed (cf. [1 ,?3]. Since planning/scheduling are currently very active. areas of research, there is no clear consensus o 1 1 which approaches are most useful. Thus, we restricted the scope of our framework to approaches that had been found useful for NASA applications in the past.

By analyzing our previous experience with building planning/scheduling systc.ins, (cf. [1 6,1'/]), as well as requirements for current and future applications, we abstracted a set of requirements that is flexible enough to support a wide. range of applications. We found that the. requirements for an planning/scheduling application framework included the following:

- A hierarchical representation of activities
- The ability to reason about: temporal Relationships between activities
- Reasoning about resource usage
- Reasoning about arbitrary state attributes

- Reasoning dependencies between vat ious parameters of activities
- 1 ·lexible representation of time.
- Support for a wide range of search algorithms
- A graphical user interface for viewing and manipulating plans and schedules interactively.
- A modeling language that is capable of expressing the reasoning facilities lisle.d above., and a parser that translates user mode.ls expressed in this language to the system's internal data structures.

Based on these requirements, we designed the components described below.

Activity Database

The c entral data structu re in ASPEN is an *activity*. An activity represents an action or step in a plan/schedule. An activity has a start time, end time, and a duration. Activities can use one or more resources. All activities in a plan/schedule are elements of *the Activity Database* (ADB), which maintains the state of all of the activities in the current plan/schedule, and serve.s as the integrating module that provides an interface to all of the other components.

One function of the ADB is to represent and maintain hierarchical relationships between activities. Activities can contain other activities as *subactivities*; this facility can be used 10 reason about the plan/schedule at various levels of abstractions (e.g., a scheduler can first reason about a set of activities without considering that each of those activities are themselves composed of a set of subactivities this can make various reasoning tasks much mom computationally tractable.

Temporal and resource-constraints between activities arc also represented in the ADB Although most of the actual computational mechanisms that maintain these constraints are implemented in other the modul es described below, the functionality that integrates constraints in the context of a plan/schedule is implemented in the. ADB. For example, although the resource timelines are responsible for detecting overuse of resources by activities, the ADB maintains data structures that indicate the. assignment of activities to specific timelines , so that one can efficiently queries such as , "which resources does this activity use?"

As another example: although the temporal constraint network (see below) is responsible for maintaining temporal constraints between individual activities, the ADB is responsible for global constraints. (e.g., the ADB contains global constraints such as: "all activities occur after the start of the scheduling horizon"; when an activity is created, the temporal constraint that the activity occurs after the horizon is created automatically by the. ADB.

Temporal Constraint Network

A Temporal Constraint Network (TCN) is a graph data structure which represents temporal constraints between activities. A temporal constraint describes the temporal relationship between an activity and other activities and/or the scheduling horizon, and impose an ordering (m the set of activities. The TCN implements a Simple. Temporal Problem, as defined in [3], and represents a set of constraints, all of which must be satisfied at any given time, i.e., it represents the conjunct of all active constraints between activities in the ADB. Activities a J c represented in the TCN as pairs of time points, where each time point corresponds to the beginning or end of an activity, and tile edges in the TCN graph represent the constraints on tile. temporal distance between the time points. The TCN can be queried as to whether the temporal constraints currently imposed between the activities are consistent or not (i. e.., whether the current ordering is free of cycles).

Resource Timelines

Resource Timelines are used to reason about the. usage of physical resources by activities. Capacity conflicts are detected if the. aggregate usage. of a resource exceeds its capacity at any given time. Several subclasses of resource timelines are implemented, including depletable resource timelines used to model consumable resources (e.g., fuel), and non-depletable resources that are used to model resources which are not actually consumed by usage, but are instead "reserved" for a period of time (e.g., a piece of equipment). Our current model of resource usage is discrete. That is, if we specify that an activity such as move-forward uses 2 units of fuel, then both of these units are modeled as being immediately consumed at the beginning of the activity. This is a discrete approximation, since the usage of the fuel may be better modeled as a linear function such as *usage(t) = t *2/(activitylength),* where *t* is the time elapsed since the beginning of the activity, and *activitylength* is the duration of the activity.

State Timelines

State timelines represent arbitrary attributes, or *states*, that can change over time.. Each state can have several possible values; at any given time, a state has exactly one of these values. Activities can either change or use states. For example, a *door-open* activity would set the state. of' door to be open, while an *enter-building* activity would require that the state of door be open. As activities are placed/moved in time, the state timeline update.s the value.s of the. state, and detects possible inconsistencies or conflicts that can be introduced as a result. For example, an activity that requires that the door be open is place.d at time *t*, then the state timeline checks to verify that the door is in fact open at time *t*. Otherwise, a state constraint violation is indicated. Users can define legal sequences of state transitions. The state timeline module will detect illegal transition sequences if they are introduced into the timeline.

For example, consider modeling a traffic light with a state. timeline, traffic-light. The possible values are *green*, *yellow*, and *red*. The legal state value transitions are: *green* to *yellow*, *yellow* to *red*, *red* to *green*. All other transitions are illegal.

Parameter Dependency Network

Each activity has a number of parameters, that are either user-defined or computed by the system, such as start time, end time, duration, any resources it uses, any states it changes/uses, etc. In ASPEN, it is possible to create dependencies between parameters, between pairs of parameters within the same activity, or between pairs of parameters defined in different parameters. A dependency between two parameters *p1* and *p2* is defined as a function from one parameter to another, $p1 = f(p2)$, where $f(x)$ is an arbitrary function whose input is the same type as p2, and whose output has the type. of p 1. These dependencies are represented and maintained in a Parameter Dependency Network (PDN). The PDN maintains all dependencies between parameters, so that at any given time, all dependency relations are satisfied. Note that if there exists a dependency such that $p1 = f(p2)$, it's inverse dependency, $p2 = f^{-1}(p1)$ dots not necessary exist, unless the user specifics the inverse relationship and specific.s the inverse dependency as well.

Planning/Scheduling Algorithms

The search algorithm in a planning/scheduling system searches for a valid, possibly near-optimal plan/schedule.

```
activity" prevalve_removal(
      duration = 15
      slot subsystem
      after prevalve_prep with (subsystem = : this.subsystem)
      before prevalve_replace with(subsystem = = this.subsystem)

      Reservation hydraulic_lift_usage {
            resource = hydraulic_ lift ;
            usage = 1 ;
            duration = 5;
      )

      requires state prevalve-purged TRUE
      requires state PYevalve- illuminated TRUE
)

Resource hydraulic_ life {
      type non- depletable
      quantity 1
)
```

Figure 1. Sample of ASPEN modeling language (part of the Reusable Launch Vehicle maintenance model). This describes an activity for removing the prevalve of an engine subsystem.

The ASPEN framework has the flexibility to support a wide range of scheduling algorithms, including the two major classes of AI scheduling algorithms: constructive and repair-based algorithms.

Constructive algorithms (e.g., [4] incrementally construct a valid schedule, taking talc. that at every step, the partial schedule constructed so far is valid When a complete schedule is constructed, it is therefore guaranteed to be valid. Repair-based algorithms(cf. [12,22]) generate a possibly invalid complete schedule using either random or greedy techniques. Then, at every iteration, the scheduled is analyzed, and repair heuristics that attempt to eliminate conflicts in the schedule are iteratively applied until a valid schedule is found.

The search algorithms that have currently been implemented include

- *forward dispatch*, a greedy, constructive algorithm,
- IRS, a constructive, backtracking algorithm based on [17]; and
- DCAPS, a repair-based algorithm based on [16].

Graphical User Interface

The ASPEN Graphical User Interface (GUI) component provides tools for graphically displaying and manipulating schedules. Resource and state. timelines are displayed. Activities are overlayed on the timelines, and users can dire.ctly manipulate activities using standard drag-and-drop procedures.

Extending ASPEN for Applications

There are two means by which ASPEN can be extended and specialized for a particular application. These arc:

- Creation of domain-specific models using the modeling language, and
- Extension of the application framework code

The modeling language is used to specify domain-specific constraints and activities. Figure 1 shows an example of part of a domain model specified in the ASPEN modeling language.

The base ASPEN framework, including the modeling language is sufficiently extensible to support a range of applications without any extensions to the code of the framework itself (e.g., the Reusable Launch Vehicle ground maintenance scheduling application is directly derived from the framework by simply specifying a model file)

Extensions to the framework code needs to be made when changes in the behavior of ASPEN components are required. This includes two Classes of extensions: *epistemological* and *heuristic*.[2] Epistemological extensions are necessary when new representational capabilities are in order to model a new domain. For example, if we wanted to implement a new type of resource timeline which had a more sophisticated, continuous model of resource usage[3], then a new subclass of the resource timeline class would need to be implemented. Heuristic extensions customize the behavior of the framework so that the behavior is more efficient for a particular domain.[4] Examples of heuristic extensions include new repair heuristics for a repair-based scheduler, or an entirely new search algorithms.

4. APPLICATIONS OF THE ASPEN SCHEDULING SYSTEM

In this section, we describe ongoing applications of the ASPEN scheduling system to: generation of spacecraft command sequences for the New Millennium Earth Observing One satellite and the U.S. Navy UHF Follow On One (UFO-1) satellite; generation of mission operations sequences to assist in design analysis for science and operability; and rapid generation of plans for maintenance and refurbishing for Highly Reusable Space Transportation.

[2] This classification follows [11]

[3] Recall from Section 3 that we use a simple, discrete resource usage model.

[4] This implies that the framework is, in principle of solving the problem without a heuristic extensions.

Spacecraft Commanding

The primary application area for the ASPEN scheduling system is generation of spacecraft command sequences from high level goal specifications. In this role, automated scheduling systems will encoding of complex spacecraft operability constraints, flight rules, spacecraft hardware models, science experiment goals and operations procedures to allow for automated generation of low level spacecraft sequences by use of advanced Artificial Intelligence planning and scheduling technology. By automating this process and encapsulating the operation specific knowledge we hope to allow spacecraft commanding by non-operations personnel, hence allowing significant reductions in mission operations workforce with the eventual goal of allowing direct user commanding (e.g., commanding by scientists).

Current efforts in applying the ASPEN scheduling system to spacecraft commanding focus on two missions: the New Millennium Earth Observing One (NM EO-1) satellite (to be launched in late 1998) and the U.S. Navy UHF Follow On One (UFO-1) satellite (currently in orbit). NM EO-1 [10] is a earth imaging satellite featuring an advanced multi-spectral imaging device. For this mission, operations consists of managing spacecraft operability constraints (power, thermal, pointing, buffers, consumables, engineering downlinks, etc.) and science goals (imaging of specific targets within particular observation parameters). Of particular difficulty is managing the downlinks as the amount of data generated by the imaging device is quite large and ground contacts are a limited resource.

Another ongoing effort in the area of spacecraft commanding is the development of an advanced commanding system for the U.S. Navy UFO-1 satellite [19]. UFO-1 is an on-orbit testbed managed by the U. S. Naval Academy Space Artificial Intelligence Lab (SAIL) at Annapolis In this collaboration, SAIL is developing an uplink, downlink, basic data transport, and commanding capability to be interfaced with an advanced planning and scheduling engine (ASPEN). In this application, ASPEN will allow high level commanding of the UFO-1 satellite to perform high level functions such as: auto pitch momentum dumping, preparation for eclipse season, delta- V maneuvers, IRU warmup and turnon, battery cell pressure bias callibration, delta inclination maneuvers, and other engine.cril)g housekeeping functions. 'Jim ASPEN scheduling engine then performs appropriate. expansion and conflict resolution to generate lower level command sequences to achieve the higher level goals.

Design Evaluation

Another application of the ASPEN system is in support of the Pluto Express (PX) project for the dual purposes of science planning and design evaluation for science and operability [14]. In support of science planning, the we are developing high level models of proposed PX spacecraft to assist in automated generation of science data acquisition plans (e.g., high level activity sequences) from high level science Seals to assist in developing science plans for mission profiling,.

In a spin-off application of planning and scheduling technology, this same capability to generate science (and engine.c]illf,) plans is being used to evaluate candidate spacecraft designs from tile. standpoint of emergent design aspects such as science return and operability. This spin-off arise.s from tile observation that often it is difficult to determine how well a given spacecraft design will perform without fleshing out approximate operations sequences for critical phases of the mission (e. g., encounter). 111 order to address this (difficulty, we are developing a design analysis tool which accepts as input: a candidate spacecraft design (and operations constraints, models, etc.); a set of engine.cring and science objectives; and a set of scoring functions to assess how well a sequence achieves the objectives. This tool then uses planning and scheduling technology to generate a candidate sequence; then uses the scoring function to score this sequence in terms of the aspects of science, operability, etc. The end result is that the design team gets quantitative feedback on how well the design achieves science objectives and meets operability constraints (such as meeting power margins, etc.). This type of tool is targeted at enabling design teams to rapidly and impartially evaluate large numbers of spacecraft designs with little effort, thus allowing improved analysis of design tradeoffs to enhance science and operations concerns for future. missions.

Maintenance Scheduling

The Highly Rc.usable Space Transportation (HRST) program targets development of technologies leading to highly reusable space transportation systems which will provide extremely low cost access to space [8,9]. As part of this program, we have been developing and demonstrating advanced scheduling systems for the rapid generation and revision of plans for maintenance and refurbishment of highly reusable launch vehicles. In this application, rca-time telemetry downlinked either during flight or immediately after flight would be analyzed to automatically generate a set of maintenance requests. These maintenance requests would then be transformed into a refurbishment plan by an automated planning and scheduling system which would account for available equipment and resources as well as the intricacies of the refurbishment procedures of the highly complex propulsion systems. The end target is to allow a turnaround of several days for the 111<S'1' space.cmft to support a flight frequency on the order of one flight per 1-2 weeks (as a comparison, the space shuttle refurbishment process takes approximately 65 days with a flight frequency of once per 4 months). In the maintenance schedule can be generated using in-flight telemetry then the refurbishment process call be speeded even fill ther by allowing for downlinking of requests for pre-positioning of equipment and resources to minimize schedule delay.

Once. the actual maintenance plan has been generated - the planning tool continues to be of use in two ways. First, in many cases there can be several mutually exclusive maintenance activities which can be performed next. Via lookahead and critical path analysis automated scheduling software can determine the next activities to enable the minimal makespan (overall schedule execution time). Second, as unexpected events arise (such as equipment failures, resource unavailabilities, and schedule slippage.), the automated scheduling software has the ability to revise the schedule so as to minimize schedule disruption (movement of activities and resources from their original assignments) and schedule slippage (delay of the completion of the overall refurbishment.

In order to test and validate this technology we have been utilizing test maintenance procedures developed by Rockwell International during the Phase 1 competition of the. Reusable Launch Vehicle Program which ended in July 1996. Specifically, we used the maintenance procedures developed for the 1.(), and 1,11, propulsion systems for the X-33 Reusable Launch Vehicle. The procedures derived for maintaining and refurbishing the test articles provided a rich testbed for Space Propulsion System Maintenance Scheduling. In all, the testbed involved refurbishement of 2 major systems with 16 subsystems, with a total of 576 lowest level activities in a complete (every subsystem) refurbishment plan. The mode] that we developed for schedule generation involved: 576 activity types, 6 resources, and on average 6 state., resource, and precedence constraints per activity. In this application we allowed maintenance requests to request either refurbishment of specific sut)sys(c.ills or major systems. In order to schedule the maintenance requests the ASPEN system used a forward sweeping greedy dispatch algorithm which used strong knowledge of (tic precedences of activities in the plan. The resultant scheduler has been able to generate schedules for refurbishment problems involving half of the subsystems (8 subsystems, 288 activities) in approximately t 0 minutes.

5. RELATED WORK

Recent reviews of planning and scheduling can be found in [I ,23]. [15] provides a comprehensive treatment of application frameworks.

DITOPS [18] is a recently developed application framework for scheduling in the domain of production management. The differences between DITOPS and ASPEN are the following:

- 111'1'01'S is designed for production management domain, while ASPEN is designed for spacecraft operations domains.
- DITOPS is a scheduling framework, while ASPEN is a hybrid planning/scheduling framework.
- DITOPS is commited to the paradigm of repair-bad scheduling; ASPEN is

uncommitted to either the repair-basd or constructive approaches.

6. CONCLUSIONS AND FUTURE WORK

interface to adaptive problem Solver's - automatic configuration

In this paper, we have described ASPEN, a reconfigurable, modular framework for planning/scheduling applications, and described three current applications of ASPEN in spacecraft operations. Although the development of a generic software architecture such as ASPEN has required a substantial, initial investment of effort, we expect the total development effort for a set of scheduling applications to be significantly decreased (as compared to individually developing each of the applications).

ASPEN is currently a scheduling-oriented system, although some planning capabilities are supported for hybrid planning/scheduling applications such as the EO-1 and UFO-1. We plan to extend ASPEN to support additional planning capabilities. Currently, ASPEN already supports much of the functionality of state of classical planning systems [1]. We plan to extend ASPEN's planning capabilities so that it can be used as a framework for powerful planning applications that also exploit the additional temporal reasoning and resource management capabilities which are available through ASPEN's scheduling-oriented facilities. ·

Finally, the reconfigurability of ASPEN could be taken a step further. Currently, it is not possible to reconfigure ASPEN at runtime. Implementing the ability to be reconfigured at runtime would enable the interesting possibility of using adaptive problem solving techniques to automate the reconfiguration process for particular c.lasses of problems (cf. [6] or for partiular problem instances (cf. [5]).

ACKNOWLEDGMENTS

This work was performed by the Jet Propulsion Laboratory, California Institute. of Technology, under contract 10 the National Aeronautics and Space Administration.

REFERENCES

[1] J. Allen, J. Hendler, A. Tate, Readings in Planning, Morgan Kaufmann, 1990.

[2] M. Deale, M. Yvanovich, D. Schnitzius, D. Kautz, M. Carpenter, M. Zweben, G. Davis, B. Daun, "The Space Shuttle Ground Processing System," in Zweben M., Fox M., ed., *Intelligent Scheduling*, Morgan Kaufman, 1994.

[3] R. Dechter, I. Meiri, J. Pearl, "Temporal constraint networks," Artificial Intelligence (49)61 -95, 1991.

[4] M. Fox, "ISIS: a retrospective," in M. Zweben and M. Fox, eds., *Intelligent Scheduling*, Morgan Kauffman, 1994,

[5] A. Fukunaga, S. Chien, D. Mutz, R. Sherwood, A. Stechert, "Automating the process of spacecraft design optimization," to appear in *Proceedings of IEEE Aerospace*, **1997.**

[6] J. Gratch, S. Chien, "Adaptive problem-solving for large-scak scheduling problems: a case study", *Journal of Artificial Intelligence Research*, (4)365-396, 1996.

[7] S. Graves, "A Review of Production Scheduling," *Operations Research*, 29(4):646-675, 1981.

[8] *Proceedings of the 1995 Technical Interchange Meeting on* ***highly*** *Reusable Space Transportation*, Huntsville, Al., July 199S.

[9] *Proceedings of the 1996 Technical Interchange Meeting on Highly Reusable Space Transportation*, Huntsville, AI,, August 1996.

[10] *Landsat-7 Mission Operations Concept Document*, Mission Operations and Data Systems Directorate, Goddard Space 1 'light Center, Greenbelt, MD, 1996.

[11] J. McCarthy, P. Hayes "Some philosophical problems from the standpoint of artificial intelligence." In B. Meltzer and D. Mitchie (eds), Machine

Intelligence 4, pp.463-502, Edinburgh University Press, 1969.

[12] S. Minton, M. Johnston., A. Philips, P. Laird, "Minimizing Conflicts: A heuristic repair method for constraint satisfaction and scheduling problems," *Artificial Intelligence*, (58)161-205, 1988.

[13] N. Muscettola, "HSTS: Integrating Planning and Scheduling," in M. Zweben, M. Fox, ed., *Intelligent Scheduling*, Morgan Kaufman, 1994.

[14] *Pluto Express FY96 Annual Report*, September 1996.

[15] W. Pree, Design patterns for object-oriented software development. ACM Press, 1995.

[16] G. Rabideau, S. Chien, T. Mann, J. Willis. S. Siewert, P. Stone, "Interactive, Repair-Based Planning and Scheduling for Shuttle Payload Operations," to appear in *Proc. IEEE Aerospace Conf.*, 1997.

[17] B. Smith., S. Chien, G. Rabideau, D. Yan, N. Muscettola., C. Fry, S. Mohan, "On-Board Planning for New Millennium Deep Space One Autonomy," to appear in *Proc. IEEE Aerospace Conf.* 1997

[18] S.F. Smith, O. Lassila "Configurable systems for reactive production management," in E. Szelke, R. Kerr, eds., *Knowledge-Based Reactive Scheduling*, International Federation of Information Processing (IFIP) Transactions B-15, 1994.

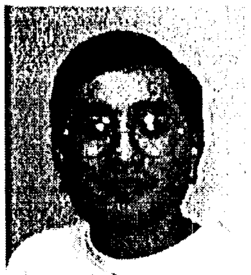
[19] *Ultra High Frequency Follow On Communications Satellite System Fact Sheet*, http://www.fafb.af.mil/uhf/fact_sheet.html.

[20] A. Weinland, E. Gamma, R. Marty, "ET++ - An Object-Oriented Application Framework in C++," in OOPSLA'88, Special Issue of SIGPLAN Notices; 23(11):46-57, 1988.

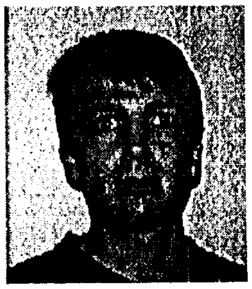
[21] D. Wilson, L. Rosenstein, D. Shafer *Programming with MacApp*, Addison-Wesley, 1990.

[22] M. Zweben, B. Daun, E. Davis, M. Deale, "Scheduling and Rescheduling with Iterative Repair," in M. Zweben, M. Fox, ed., *Intelligent Scheduling*, Morgan kaufman, 1994.

[23] M. Zweben, M Fox, eds. *Intelligent Scheduling* Morgan Kaufmann, 1994.

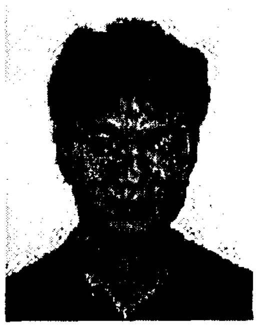


Alex S. Fukunaga *is a Member of the Technical Staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory, California Institute of Technology. He holds an A.B. in Computer Science from Harvard University, and a M.S. in Computer Science from the University of California at Los Angeles, where he is currently a Ph.D. student. His research interests include optimization, decision theory, search, machine learning, and automated planning/scheduling,*



Gregg Rabideau *is a Member of the Technical Staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory, California Institute of Technology. His main focus is in research and development of planning and scheduling systems for automated spacecraft commanding. Other projects include planning and scheduling for the first deep space mission of NASA's New Millinium Program, and for spacecraft design of the Pluto Express project. Gregg holds a B.S. and M.S. degree in Computer Science from the University of Illinois where he specialized in Artificial Intelligence.*

Steve Chien *is Technical Group Supervisor of the Artificial Intelligence Group of the Jet Propulsion Laboratory, California Institute of Technology where he leads efforts in research and development of automated planning and* *scheduling systems. He is also an adjunct assistant professor in the Department of Computer Science at the University of Southern California. He holds a B.S., M.S., and Ph.D. in Computer Science from the University of Illinois. His research interests are in the areas of: planning and scheduling, operations research, and machine learning.*



David S. Yan *is a Member of the Technical Staff in the Artificial Intelligence Group at the Jet Propulsion Laboratory, California Institute of Technology. He holds a B.S. in Electrical Engineering and Computer Science from the University of California at Berkeley. He will be pursuing his M. S. degree in Computer Science at Stanford University. His research interests include automated planning/scheduling, operating . systems, computer architecture and computer net works.*